**Example2: Stored Procedures**

**Scenario1:** PL/SQL procedure block for Monthly interest processing for savings accounts

**Objective:** Create a stored procedure block processMonthlyInterest to calculate and update the balances of all the savings accounts by applying 1% interest rate to the current balance.

**Assumptions for Table Creation:**

* Table: accounts
* Columns: account\_id, account\_type, balance
* Savings accounts are identified with account\_type = ‘SAVINGS’

**Procedure Logic:**

* Firstly, loop through the accounts table with account\_type = ‘SAVINGS’.
* Then, for each calculate the balance i.e.

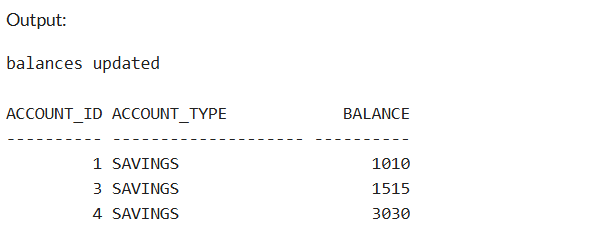
balance \* 1.01

* Finally update the account with new balance.

**Construction of stored procedure PL/SQL block:**

monthly\_interest.sql

|  |
| --- |
| CREATE TABLE accounts (  account\_id NUMBER PRIMARY KEY,  account\_type VARCHAR2(20),  balance NUMBER  );  INSERT INTO accounts VALUES (1, 'SAVINGS', 1000);  INSERT INTO accounts VALUES (2, 'CURRENT', 2000);  INSERT INTO accounts VALUES (3, 'SAVINGS', 1500);  INSERT INTO accounts VALUES (4, 'SAVINGS', 3000);  CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest  IS  BEGIN  UPDATE accounts  SET balance = balance \* 5  WHERE account\_type = 'SAVINGS';  dbms\_output.put\_line ('balances updated');  END;  /  BEGIN  ProcessMonthlyInterest;  END;  /  select \* from accounts where account\_type = 'SAVINGS'; |



**Scenario2:** Employee Bonus Scheme for employees based on Department

**Objective:** To create a store procedure program i.e. updateEmployeeBonus to increase the salary of employees in a specific department by bonus percentage.

**Assumptions for Table Creation:**

* Table: Employees
* Columns: emp\_id, dept\_id, salary.
* Inputs of table:
* update\_dept\_id: Department to update
* update\_bonus: Bonus percentage

**Logic of Procedure:**

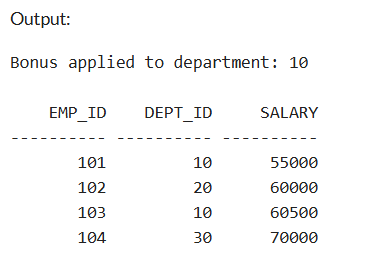
* The parameters are dept\_id and bonus\_percentage.
* Calculate: updated\_salary = salary + (salary \* bonus / 100)

The updated salary is to be calculated for each department.

**Construction of PL/SQL procedure block:**

emp\_bonus.sql:

|  |
| --- |
| CREATE TABLE employees (  emp\_id NUMBER PRIMARY KEY,  dept\_id NUMBER,  salary NUMBER  );  INSERT INTO employees VALUES (101, 10, 50000);  INSERT INTO employees VALUES (102, 20, 60000);  INSERT INTO employees VALUES (103, 10, 55000);  INSERT INTO employees VALUES (104, 30, 70000);  CREATE OR REPLACE PROCEDURE updateEmployeeBonus (  update\_dept\_id IN NUMBER,  update\_bonus IN NUMBER  )  IS  BEGIN  UPDATE employees  SET salary = salary + (salary \* update\_bonus / 100)  WHERE dept\_id = update\_dept\_id;  DBMS\_OUTPUT.PUT\_LINE ('Bonus applied to department: ' || update\_dept\_id);  END;  /  SET SERVEROUTPUT ON;  BEGIN  updateEmployeeBonus (10, 10);  END;  /  SELECT \* FROM employees; |



**Scenario3:** Transfer funds between the accounts.

**Objective:** To create a procedure block i.e. transferFunds to transfer funds between two accounts and make sure that the source account has sufficient balance before transfer.

**Assumptions for Table Creations**:

* Table: accounts
* Columns: acc\_id, balance.
* Inputs to be taken:
* src\_id: source account
* dest\_id: Destination account/ Target account.
* amt: amount to transfer

**Procedure Logic:**

* Initially, fetch the existing balance of the source account.
* If balance is greater or equal to transfer / destination account then
* Subtract amount from source account
* Add the amount to the destination/target account.
* Or else,
* Raise an exception or display an error message.

**Construction of PL/SQL procedure block:**

transfer\_funds.sql

|  |
| --- |
| CREATE TABLE accounts (  account\_id NUMBER PRIMARY KEY,  balance NUMBER  );  INSERT INTO accounts VALUES (101, 5000);  INSERT INTO accounts VALUES (102, 3000);  INSERT INTO accounts VALUES (103, 1000);  select \* from accounts;  CREATE OR REPLACE PROCEDURE transferFunds (  src\_id IN NUMBER,  dest\_id IN NUMBER,  amt IN NUMBER  )  IS  v\_source\_balance NUMBER;  BEGIN  SELECT balance INTO v\_source\_balance  FROM accounts  WHERE account\_id = src\_id;  IF v\_source\_balance >= amt THEN  UPDATE accounts  SET balance = balance - amt  WHERE account\_id = src\_id;  UPDATE accounts  SET balance = balance + amt  WHERE account\_id = dest\_id;  DBMS\_OUTPUT.PUT\_LINE ('Transfer of Rs. ' || amt || ' completed from ' || src\_id || ' to ' || dest\_id);  ELSE  RAISE\_APPLICATION\_ERROR (-20001, 'Maintain enough balance in source account.');  END IF;  END;  /  SET SERVEROUTPUT ON;  BEGIN  transferFunds (101, 102, 2000);  END;  /  SELECT \* FROM accounts; |

Expected output:

